HD View: A Software Camera for the Web

Matt Uyttendaele Howard Good Michael F. Cohen
Microsoft Research
MSR-TR-2009-95

Abstract

As captured imagery grows in resolution to gigapixels, increases in dynamic range beyond what can be displayed, and encompasses wider fields of view, there is a need for the viewer of such imagery to take on many of the roles traditionally relegated to the camera. Fully exploring such imagery requires a software camera*. One such software camera is HD View. We discuss the need for the evolution towards software cameras and then describe the current implementation of HD View. HD View is an interactive software camera that provides flexible output lens control, full color management, and interactive tone mapping of high resolution, wide gamut, and high dynamic range imagery.*

1. Introduction

As digital photographs grow ever larger in resolution, deeper in bit depth, and wider in field of view, the disparate roles of the device that captures the light and the software used to organize and display the results have begun to shift. This need for a shift becomes more apparent as image stitching software results in gigapixel sized images and newer monitors are capable of displaying high dynamic range and wider color gamuts. Less that one thousandth of the resolution of multi-gigapixel sized images can be displayed at any moment and the mapping of radiance and color to output pixels becomes dependent both on the monitor and the portion of the scene being rendered. Thus the role of what we will term a *software camera* for viewing the imagery takes on more complexity and importance. HD View is the latest in a history of such *software cameras*. After adding some depth to the paragraph above, we will discuss some of the capabilities and implementation details of the current version of HD View. Finally, we discuss the implications for the future for both capture devices and software cameras.

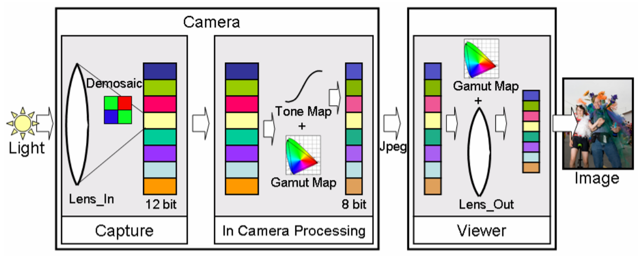


Figure 1. Photography Today

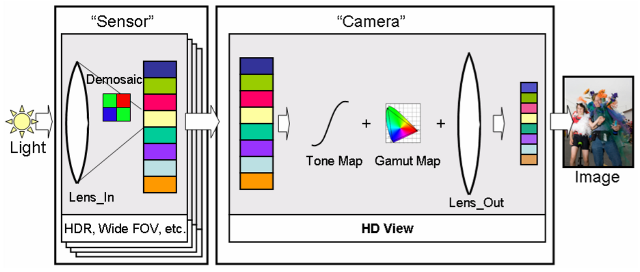


Figure 2. Photography with a Software Camera

1.1. Photography Today

Let's begin with a simplified look at what happens when we take a picture (see Figure 1). A camera allows a set of photons to enter it. A sampling of this light is integrated over a small solid angle (a ray), over some (typically small) amount of time, and over some portion of the visible spectrum resulting in a scalar value at each sensor location. The directional integration is defined by a lens, the temporal integration by a shutter, and the spectral integration by a filter. Because of the ways sensors are designed a de-mosaicing step combines nearby spectral samples into pixels lying in a tri-chromatic color space. The resulting scalar tristimulus values for each *ray* represent the image in the camera.

Given the set of ray values, processing is typically performed to transform to some (usually *output-referred*) color space, nonlinearly compressed to 8 bits (tone mapping), and filtered for other effects (e.g., sharpening, contrast enhancement, etc.). The result is often a JPEG image.

Finally, a *viewer* maps the pixels to an output device. We will describe the mapping from input values which represent rays to the output pixels as an *output lens*, for reasons that become clearer later. Most often, this lens simply scales and filters an input array to fit some rectangular area of the screen, but there is no reason to be constrained to such a simple mapping.

1.2. Photography with a Software Camera

A similar set of steps from input light to output image explains why the *viewer* should evolve to become a *software camera*. The most obvious difference between Figures 1 and 2 is the high level grouping of the imaging stages. In Figure 2, the initial stage is limited to collecting samples of the light rays entering the *sensing device*. The use of the word *camera* has been deliberately avoided. The sensing device certainly might be a traditional camera recording light impinging on a rectangular array of sensors. But it might also be a set of captured sensor arrays pointing in different directions designed to be combined into a wide angle panorama, or a set of exposures over the same field of view to create a higher dynamic range result, or both. We deliberately will skip how such multiple captures are combined into a wide angle and/or high dynamic range data set as there have been multiple papers on such topics. A renderer within a virtual environment could be considered another *sensing device* that delivers radiance values arriving along a set of rays. In it most general form, the first stage in Figure 2 delivers a discretized plenoptic function to the software camera.

The set of radiance values is then delivered to the software camera for processing and display. For now, we will limit the discussion to a slice of the plenoptic function at a given point in time and at a specific point in space. Thus we will consider only the 2D set of rays passing through a point. Preferably, the incoming values are stored in some *scene-referred* color space, such as Kodak's ProPhoto gamut which is not already limited to the *output referred* sRGB color space defined to fit monitor standards. Similarly, at this stage, the color values have as much accuracy as the sensor can provide.

Given a slice of the plenoptic function, an interactive software camera then acts much like a real camera. Under user control, one can point the camera and zoom in and out to select a portion of the scene to view. The specific mapping from directions to output is defined by the output lens. This is not limited to a standard pinhole plus flat film plane lens model, but can mimic a panoramic camera with a cylindrical film surface, perform a spherical mapping or mimic a fisheye or other lens model. Much like a real camera, the *exposure* level can be set to automatically adjust to what is being viewed. Somewhat beyond hardware camera capabilities, a nonlinear mapping from radiance to output, (i.e., a compressive tone mapping operator), can be invoked to compress a high dynamic range to a lower displayable dynamic range. Similarly, areas with little contrast, (i.e., lower dynamic range than the output device), can be boosted to have their contrast increased. Finally, the mapping to output colors can leverage the capabilities and color profile of the monitor.

In the remaining sections, after covering a bit of the history of software cameras, we will discuss details of how each of the stages of our software camera, HD View, is designed and implemented. We would encourage the reader at this point to experiment with HD View at: `http://research.microsoft.com/ivm/HDView/HDR.htm`. We will end with some thoughts looking forward.

2. The Viewer as a Camera is Not New

The notion of the online viewing program acting somewhat like a camera is of course not new. Perhaps the first such *software camera* was Quicktime VR [3, 2] which introduced us to the idea of panning and zooming on spherical panoramas stored as a cube map. More recently, we have seen Zoomify [18] and other Flash based viewers such as at the Gigapan site [8] appearing for viewing high resolution panoramic imagery. However, none of these leverage the full capabilities that can be exercised at viewing time. Quicktime VR uses a perspective output lens only that leads to severe distortions at wide fields of view, while other gigapixel image viewers rely on the input mapping of rays to pixels to define the output lens. In other words if a panorama has been mapped using cylindrical coordinates, the user can only choose a sub-rectangle of the input array at any one time by panning and zooming.

Given a set of ray values not restricted to those through a single point (i.e., a lightfield) requires other mechanisms to explore the data. The original Lightfield [11] and Lumigraph [9] papers show how to efficiently view such datasets when dense. The recently introduced Lightfield Camera [15] discusses refocusing as well by simulating a large aperture. The Photo Tourism work [17] showed how multiple photographs taken from varying points of view could be organized and viewed interactively by navigating through the sparse 5D lightfield between the dense clusters (the photographs themselves). Fully viewing these more general data sets is currently beyond the capabilities of HD View.

Although each of the above instances of software cameras provides some form of pointing, zooming, and positioning the camera, none of these viewers provides more general output lenses, performs interactive tone mapping or color management that can adapt to monitor capabilities.

Tone mapping operators to map high dynamic range data to a displayable dynamic range are an important aspect of software cameras [13, 6, 5, 7]. For our purposes we need

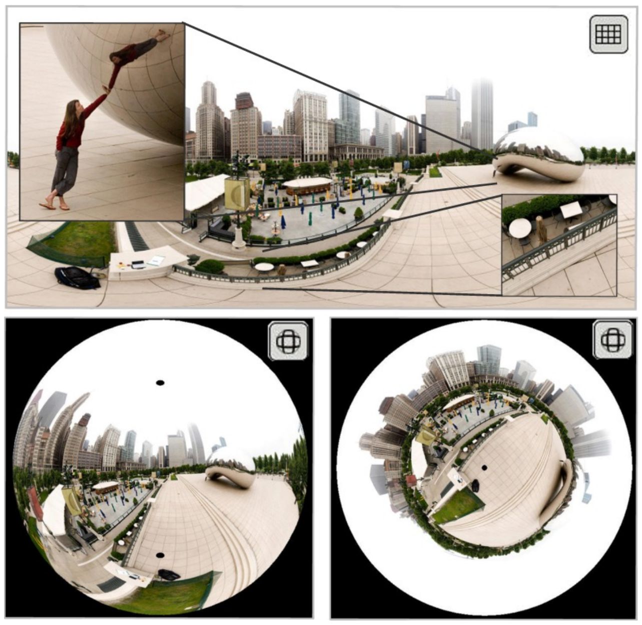


Figure 3. Top - a spherical output lens rendering of a full sphere of directions. Insets - perspective output lenses when zoomed in. Note that the curved lines of the terrace are now straight. Bottom - two views using a fisheye output lens. See `http://www.xrez.com/hdview/chi7.html`. Another interesting spherical data set can be seen at `http://digitalmedia.org.mk/miso/HDView/2008-04-13_Kombe_pano_BW.htm`.

a simple global operator that can be used in real-time. We adopt Reinhard et al's work [16] in part for compressing high dynamic range input.

3. HD View

The recent paper, Capturing and Viewing Gigapixel Images [10], discusses the capture of very high resolution panoramic images as well as a viewer that exhibits some of what we discuss here. The viewer evolved into a browser based application called HD View. As in all the gigapixel viewers, for efficiency the data is stored and fetched from a hierarchical quadtree pyramid of values. Conceptually, we can ignore this aspect of the complete system as we look at some of the newer features and implementation details since the initial paper that appeared at SIGGRAPH.

3.1. Output Lenses

When capturing imagery, the lens directs photons arriving from a ray direction to a particular sensor location. It should be noted that it is really the combination of the lens shape and the sensor geometry that determines the mapping from input ray to sensor in physical systems. For example, the same glass lens with a flat sensor back vs. a curved sensor surface can result in either a perspective or cylindrical mapping. In much the same spirit as a physical system, the software output lens in HD View directs recorded input rays screen locations.

HD View takes as input a rectangular array (really a hierarchical pyramid) of sensor data as well as information about the mapping between this array and ray directions. In other words, the data array may be parameterized by (x, y) planar coordinates as in a perspective projection, or it may be organized in cylindrical, (θ, y), or spherical, (θ, ϕ), coordinates. Similarly, the *output lens* may simulate a perspective or curved projection between ray directions and the screen coordinates. Given the data array and its mapping to rays, an output lens and view direction, HD View constructs and displays the image on the fly. Thus, there is a decoupling of the projection used to organize the input data and the projection implemented by the output lens.

Even if the input data is already in the form of a perspective mapping, a perspective output lens does not simply select a sub-rectangle for display. Rather, since rotating the camera changes the *film plane* normal, the data array is tranformed via a homography providing the sense of *looking around*.

As the output field of view widens, perspective projections suffer from severe distortions, thus HD View smoothly reshapes the output lens on the fly interpolating between a perspective and curved projection. HD View also supports a fisheye output lens (See Figure 3).

3.1.1 Implementing the Output Lens in the GPU

The implementation and use of the GPU in HD View differs significantly from the implementation in [10]. HD View's output lens model is implemented by first building a virtual surface proxy to hold the input array. For example, if the input is organized in spherical coordinates, a unit sphere surrounding the origin is constructed. Cylindrical input leads to unit radius cylinders and a perspectively organized input results in a planar surface proxy. In each case the proxy is tessellated by first tessellating the input array and projecting the vertices to the proxy. The (u, v) texture coordinates of the proxy vertices thus map back to a uniform grid on the input array.

Given the tessellated proxy surface and associated texture coordinates, the output lens defines a virtual camera centered at the origin. A perspective output lens thus simply renders the tessellated proxy surface using the texture coordinates at each vertex and interpolating the texture on the GPU. Spherical, cylindrical, and fisheye output lens require one additional step. Each vertex position on the tessellated proxy lies at some (x, y, z) representing a ray direction. This (x, y, z) is first transformed to (θ, ϕ), (θ, y), or (r, θ) coordinates for spherical, cylindrical, or fisheye outputs respectively. The output lens' orientation and projection define the screen location for the transformed coordinates. These screen locations and associated texture coordinates define textured triangles that are rendered by the GPU. A smooth transition from perspective, at fields of view below 60 degrees, to curved projections above 120 degrees, by interpolating the screen coordinates defined by the two projections. Although the linear texture interpolation within the triangles is an approximation, this implementation is simple and is flexible enough to encompass any output lens that can define a mapping from ray direction to the output lens coordinate system. In practice, by tesselating the original data array into small triangles, we have never noticed any artifacts of the approximation.

3.2. Color Management

HD View also provides full color management using greater than 8bit/component precision. This allows data to be stored in a wide gamut color profile such as ProPhoto. By operating on wide gamut *scene-referred* data, it is possible at display time to render the image optimizing for the color profile of the display. This is important for two reasons. First, we are seeing many manufacturers offering wide gamut displays using multicolored LED backlights, able to reproduce significantly more of the color gamut than the traditional, relatively impoverished, sRGB standard used for most common image formats to match older monitors. Second, HD View is performing tone-mapping on the fly (as described in the next section). It is highly desirable for the tone-mapping to operate on linear scene-referred data rather than on *output-referred* imagery such as sRGB encoded data. Thus, the HD View pipeline performs the display color management as the last step in the rendering process after all other tone-adjustment is performed.

When displaying on an sRGB monitor, the wider gamut data is not visible as it is clipped to the monitors color profile. It is thus hard to show the result here, nor have you see the difference if you do not have a wide gamut display. Figure 4 shows an approximate subjective impression of the differences one might see.

3.3. Tone Mapping

Perhaps the most striking thing one expects from a camera but not from an online viewer is the ability to change exposure on the fly as one points the virtual camera at brighter and darker regions of the scene. In the gigapixel paper [10] we discuss a method to add a gain and bias to enhance the local regions being viewed. This primarily provides a way to stretch the contrast in low contrast regions, and shift light or dark regions towrds middle gray. Since both the input and output were 8 bit the previous work did not deal with high dynamic range (HDR) to low dynamic range (LDR) compressive tone mapping. HDR encodings such as OpenEXR [12] and HD Photo that is currently under review as a Jpeg standard (JPegXR) [14, 4] are capable of storing many more than the 8 bits of the final display, thus the software camera must perform either tone compression or contrast enhancement depending on the dynamic range of the local portion of the scene being viewed.

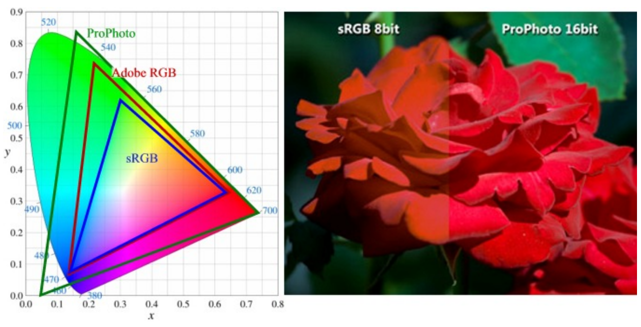


Figure 4. Left shows the range of the different color gamuts superimposed in CIE xy chromaticity coordinates. Right: a hand created image to give the subjective impression seen on a wide gamut monitor when the left half is projected into the sRGB gamut before mapping to the display. If you do have a wide gamut display you can see the example at: http://research.microsoft.com/ivm/HDView/ColorExamples.htm.

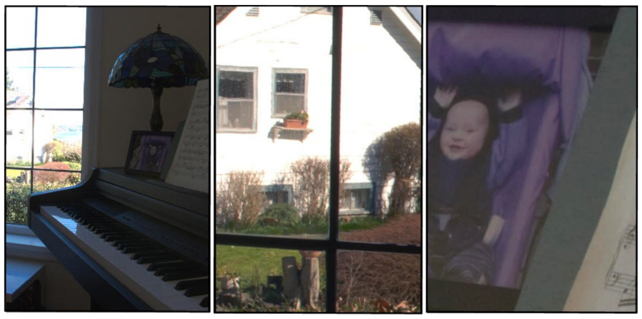


Figure 6. Linear Exposure Mode: Left - Overall image. Center, Right - Zoomed in views. (See http://research.microsoft.com/ivm/HDView/HDRExamples.htm)

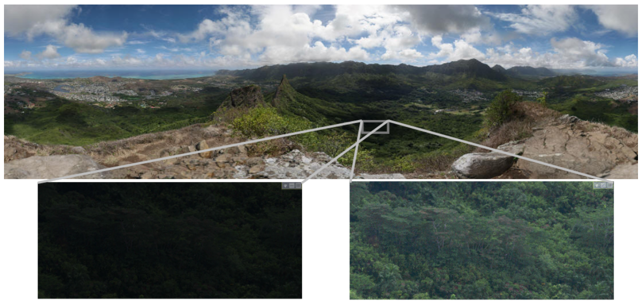
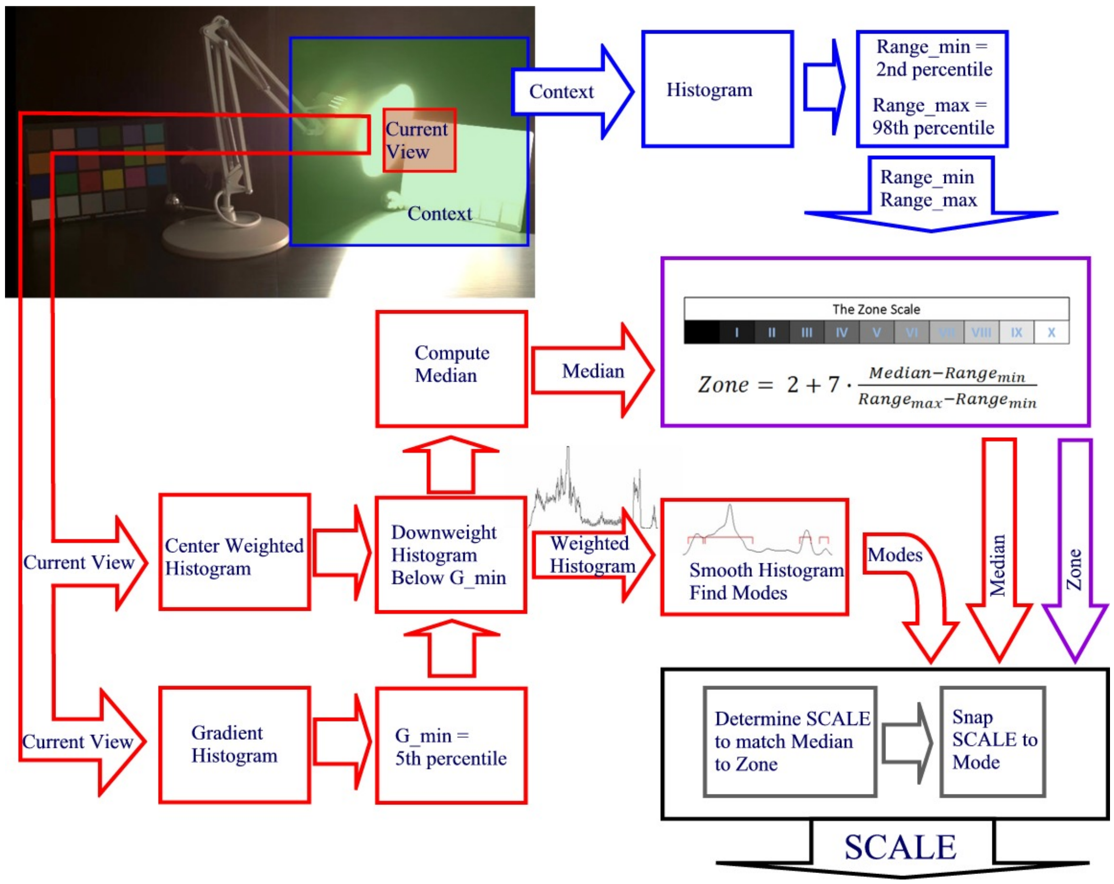


Figure 7. 360 Gigapixel Panorama. Below: detail when zoomed in without and with linear exposure activated. Panorama courtesy of xRez.(See http://www.xrez.com/hdview/index.html)



Current View
Context
Context
Histogram
Range_min = 2nd percentile
Range_max = 98th percentile
Range_min
Range_max
The Zone Scale
I II III IV V VI VII VIII IX X
Zone = 2 + 7 · (Median−Range_min)/(Range_max−Range_min)
Compute Median
Median
Current View
Center Weighted Histogram
Downweight Histogram Below G_min
Weighted Histogram
Smooth Histogram Find Modes
Modes
Median
Zone
Current View
Gradient Histogram
G_min = 5th percentile
Determine SCALE to match Median to Zone
Snap SCALE to Mode
SCALE

Figure 5. A data flow diagram for the frame by frame determination of the scale value to automatically adjust the exposure of the software camera.

HD View operates on either 8 bit Jpeg or 16 bit half-float JpegXR HDR input, and provides three tone mapping options. The first is a *fixed mode* in which the author specifies the value to be mapped to one (full brightness) and lesser values are linearly scaled between zero and one (higher values are clipped). The second and third modes perform a linear exposure or non-linear compressive tone mapping respectively with parameters determined dynamically at each frame time.

3.3.1 Linear Exposure Mode

The second mode is *auto exposure* in which the radiance value to be mapped to one (i.e., the exposure) is determined at frame time and all lesser radiance values are linearly scaled. We explain the algorithm to determine the exposure at runtime from the bottom up. All computation is run on linearized radiance values as outlined in Figure 5. Some results are shown in Figures 6 7.

The first step is to determine the range of radiance values in the surrounding *context* of the current view. We evaluate the histogram of the data in an area approximately 9x9 times the size of the current area in view. The 2nd and 98th percentile values are recorded as $range_{min}$ and $range_{max}$. We also compute a separate histogram of the values inside the current view. This histogram is center weighted (by $\cos^4$ of the distance from the center to the edge of the current window) to keep the histogram change smooth as the user pans over the scene. Finally, we compute a third histogram of the absolute value of gradients (neighboring input value differences) to help determine a lower value, G_{min}, below which we might expect to see compression and other artifacts (since we wish to accommodate compressed input). G_min is set to the 5th percentile of the gradients. Note that G_{min} will be larger in noisy and heavily discretized captures. G_{min} is used to down weight the darker end of the current view's histogram by a factor of $1 - \exp -((l/G_{min})/20)^2$.

The median value of the weighted histogram along with the range values from the context determine a target *zone* based on the zone system first derived by Ansel Adams [1] (see Figures 5). An initial *scale* value is determined that maps the median to the target zone. The *scale* value, in turn,

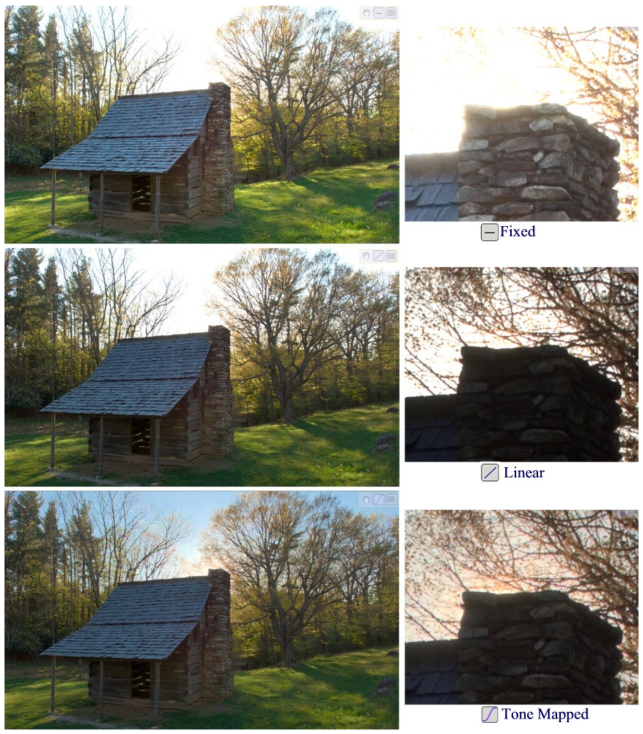


Figure 8. Top: Fixed Exposure and detail when zoomed in. Middle: Linear Exposure mode. Bottom: Tone Mapping mode. HDR data courtesy of Mark Fairchild. (See http://research.microsoft.com/ivm/HDView/HDRSurvey.htm)

determines a radiance value, L_{max}, that will be mapped to full brightness, with any radiance higher than this clipped. We found that this initial scaling sometimes led to coherent objects in the scene having parts that were clipped. Large coherent objects in the scene contribute to local clumps in the histogram. Thus, we want to avoid having L_{max} land in the middle of such a clump. To avoid this, we compute the modes in the histogram by first smoothing it and define modes to exist between any two minima containing at least 1 percent of the total histogram. If L_{max} based on the initial $scale$ falls inside a mode, then $scale$ is adjusted to place L_{max} at the top of that mode, (i.e., at a local minimum in the smoothed histogram).

This results in a linear $scale$ for the current frame that maps radiance to pixel values. One last modification avoids sudden changes to the scale value. Hysteresis is invoked to provide gradual transitions in the final image exposure as the $scale$ value changes. Specifically, $finalscale_t = finalscale_{t-1} * (scale/finalscale_{t-1})^((T_t - T_{t-1})/0.75)$, where T represents clock time in seconds, and t and $t-1$ indicate the current and previous frame. This averaging between the current and previous values of $scale$ is implemented in log space for efficiency.

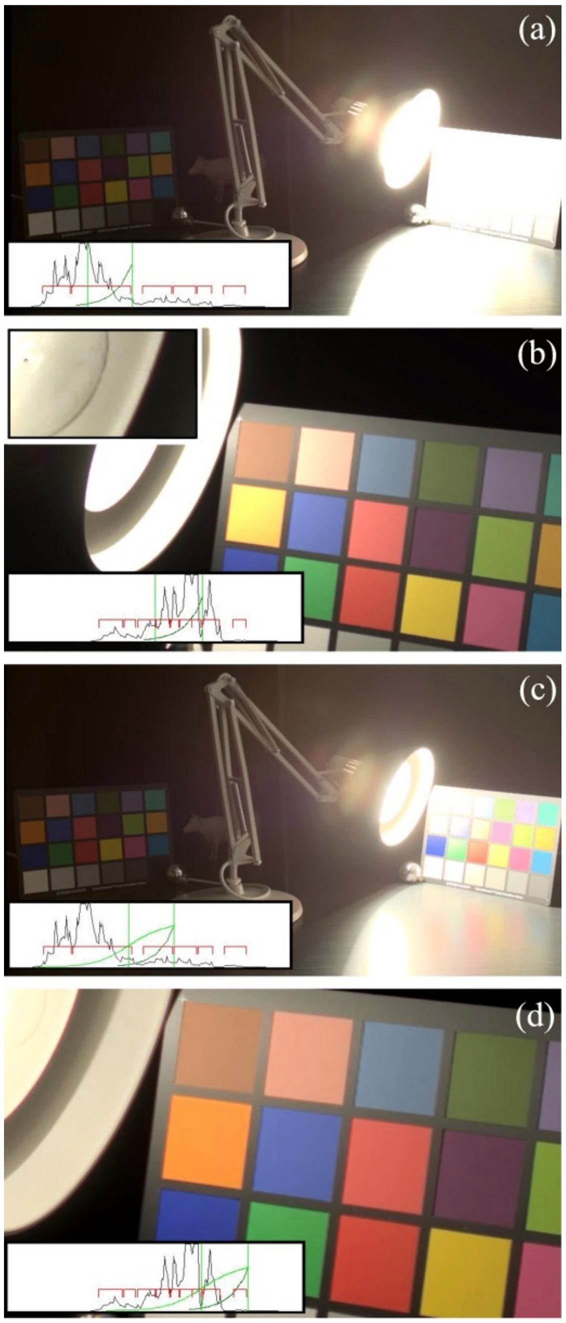


Figure 9. (a,b)Linear Exposure. (b) Note one must zoom all the way in to the light bulb to see it (inset). Bottom left of each image shows histogram, modes under red brackets, and L_{max} value mapped to full brightness (upper green bar). Note that L_{max} has snapped to mode. (c,d) Nonlinear compressive tone mapping. HDR data courtesy of Mark Fairchild. (See http://research.microsoft.com/ivm/HDView/HDRSurvey.htm)

3.3.2 Compressive Tone Mapping

While the simple linear exposure mode acts much like a camera by setting a linear mapping between radiance and output, a more general non-linear tone mapping operator can choose to bring a larger range of radiance values within the monitor's dynamic range. Similarly, areas with low contrast can be boosted by adding a bias as well as an exposure gain. The third exposure HD View does both of these within the context of creating a global tone operator.

Beginning with the contrast boosting, a boost value, c, is determined by the *range* values from the simple exposure control, such that $c = 1 + 1.5 \exp -20(log(range_{max}) - (log(range_{min})$. The radiance values, L, are then modified by pushing values away from the median by $L = c(L - median) + median$. Note that c is approximately 1.0 for any regions where there is more than a minimal range and thus the radiance values are unchanged. The *scale* is determined as before, however without the final snap to the modes and the radiances are thus scaled resulting in an L_{max} which would have been mapped to full brightness. We then snap L_{max} upward to the top of a mode that is up to a maximum of five octaves higher. Finally, we turn to Reinhard's tone mapper [16] to produce a nonlinear tome mapping $L = L(1 + (L/L_{max})/(1 + L)$ (see Figure 9).

4. Results

HD View has been downloaded by hundreds of thousands of users. HD View data sets have appeared on the front pages of leading news sites online. That said, the most recent enhancements described above have their greatest impact data sets of both high resolution and high dynamic range. Unfortunately, few of these exist at the moment although as the capabilities of cameras and image stitchers continues to increase we expect more such data sets to appear. We encourage the reader to try HD View and in particular to view some of the HDR data sets created by Mark Fairchild of the University of Rochester. A good starting point is `http://research.microsoft.com/ivm/HDView.htm`.

5. Looking Forward

Hopefully, we have provided sufficient motivation to inspire the reader to appreciate the increased role a software camera such as HD View can play in how we experience captured imagery. Even a single photograph with a modern SLR can deliver 24 Mpixels with 12 bits of accuracy, which greatly exceeds what can be displayed no a monitor. Thus all photographs can take advantage of a software camera at display time.

HD View was designed to run on low end GPUs which led to the frame-by-frame global tone mapping operations. More advanced tone mapping that responds to local regions may be possible when targeting more modern GPUs.

There are a number of additional features we would like to see in future software cameras. Lightfield cameras [15] provide the set of rays through a finite aperture, thus refocusing should certainly be a part of future software cameras. As the samples of the plenoptic function become even more ubiquitous, navigating the camera center will take on more importance. Looking ahead, we can expect the ubiquity of digital cameras and the increasing capabilities of the hardware to lead to needs for more creative designs for software cameras.

References

[1] A. Adams. *The Negative: Exposure and Development. Ansel Adams Basic Photography Series/Book 2*. New York Graphic Society, Boston, 1948. 5

[2] Apple. Quicktime vr. http://www.apple.com/quicktime/technologies/qtvr/, 2008. 2

[3] S. E. Chen. Quicktime vr: an image-based approach to virtual environment navigation. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 29–38, New York, NY, USA, 1995. ACM. 2

[4] B. Crow. Hd photo blog. http://blogs.msdn.com/billcrow/, 2008. 4

[5] F. Durand and J. Dorsey. Interactive tone mapping. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 219–230, London, UK, 2000. Springer-Verlag. 2

[6] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 257–266, New York, NY, USA, 2002. ACM. 2

[7] R. Fattal, D. Lischinski, and M. Werman. Gradient domain high dynamic range compression. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 249–256, New York, NY, USA, 2002. ACM. 2

[8] Gigapan. Gigapan. http://www.gigapan.org/about.php, 2008. 2

[9] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, New York, NY, USA, 1996. ACM. 2

[10] J. Kopf, M. Uyttendaele, O. Deussen, and M. Cohen. Capturing and viewing gigapixel images. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, 26(3), 2007. 3, 4

[11] M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on*

Computer graphics and interactive techniques, pages 31–42, New York, NY, USA, 1996. ACM. 2

[12] I. Light and Magic. Openexr. Industrial Light and Magic, http://www.openexr.com/, 2008. 4

[13] D. Lischinski, Z. Farbman, M. Uyttendaele, and R. Szeliski. Interactive local adjustment of tonal values. *ACM Trans. Graph.*, 25(3):646–653, 2006. 2

[14] Microsoft. Hd photo. http://www.microsoft.com/ windows/windowsmedia/forpros/wmphoto/, 2008. 4

[15] R. Ng, M. Levoy, M. Brdif, G. Duval, M. Horowitz, and P. Hanrahan. Light field photography with a hand-held plenoptic camera. Technical report, April 2005. 2, 7

[16] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. Photographic tone reproduction for digital images. *ACM Trans. Graph.*, 21(3):267–276, 2002. 3, 7

[17] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, 2006. 2

[18] Zoomify. Zoomify. http://www.zoomify.com/, 2008. 2

Some notes on using HD View

As stated above, HD View currently runs only under windows, but can run in all major browsers. You will be asked to install the HD View plug-in the first time you visit an HD View data set. The images can be panned using the left mouse or arrow keys. Zooming is easiest using a mouse wheel but can also be done by double clicking (left and right) or with the + and - number pad keys. The upper right has three icons for mode switching. From left to right these are for switching the UI for panning, the tone mapping mode, and the output lens. See `http://research.microsoft.com/ivm/HDView/HDHelp.htm` for more details.